9th UCAAT User Conference on Advanced Automated Testing

Systematic Selection of Testing Methodology for Low-Code Development

Shreyasi Warunkar, Baris Güldali

CQM
S&N GROUP

15/09/2022

Introduction

Low-Code Development & Testing

- What is Low-Code?
- Testing Low-Code applications

Complexity of Low-Code applications

- Case studies
- Complexity characteristics

Test Methodology for Low-Code

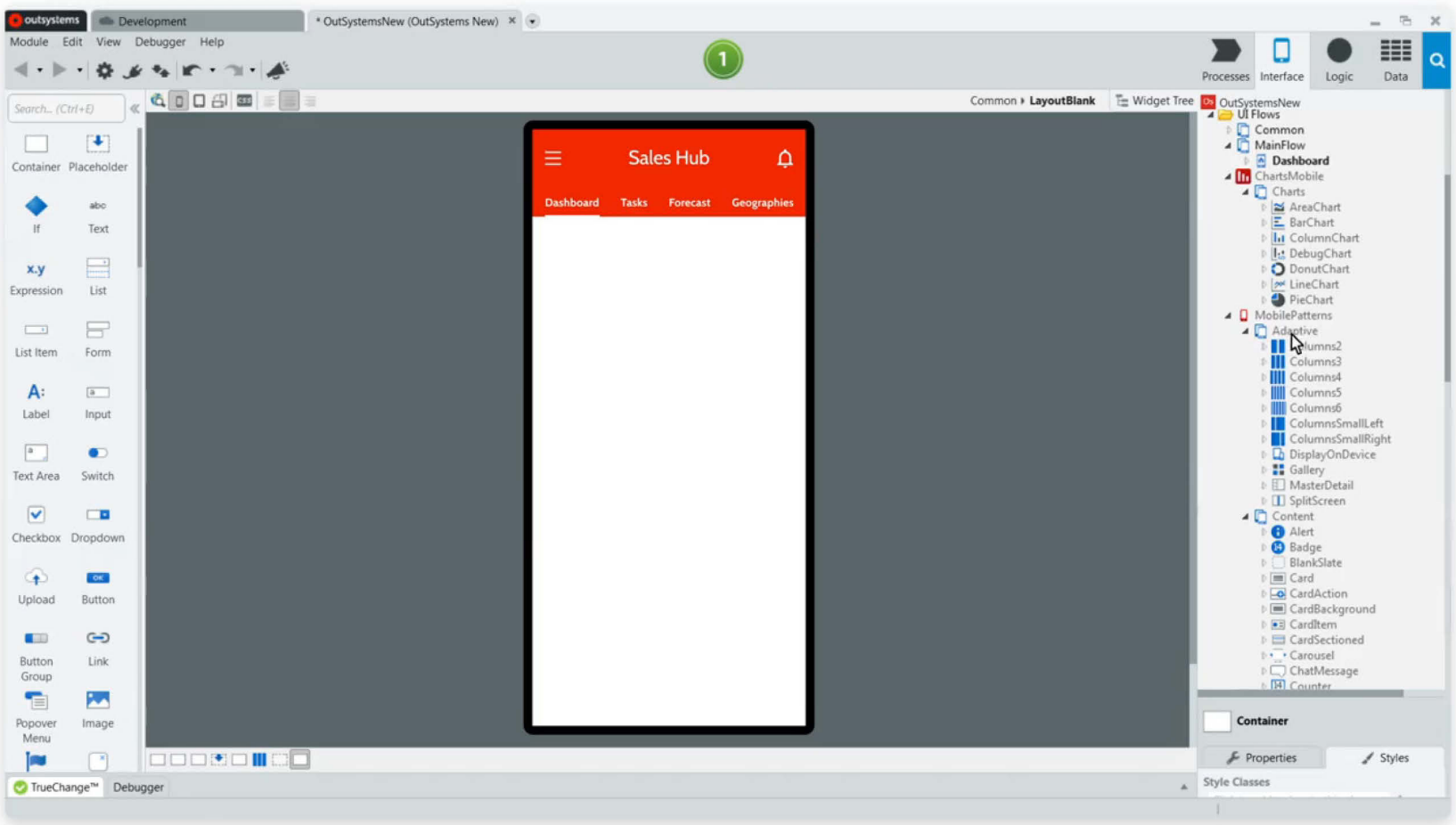
Low-Code Development & Testing

What is Low-Code?

Testing Low-Code applications

outsystems mendix appian

- Data Modeling
- User Interface Design
- Specification of busines logic rules and workflows
- Integration of external services via third-party APIs
- Application Deployment



Complexity of Low-Code applications

Case studies selected = 100 (Outsystems = 50, Mendix = 50)

Cases from variety of application domains –

Healthcare, Finance, Logistics, Insurance, Government, NGO, IT etc.

Complexity characteristics:

- Application type
- Time to develop
- Need of training
- Integrations with other systems
- Scalable
- Safety critical
- Customized
- Testing/QA mentioned
- Agile methodology incorporated

Source: https://www.featuredcustomers.com/
https://www.outsystems.com/case-studies/
https://www.mendix.com/customer-stories/

Development Time

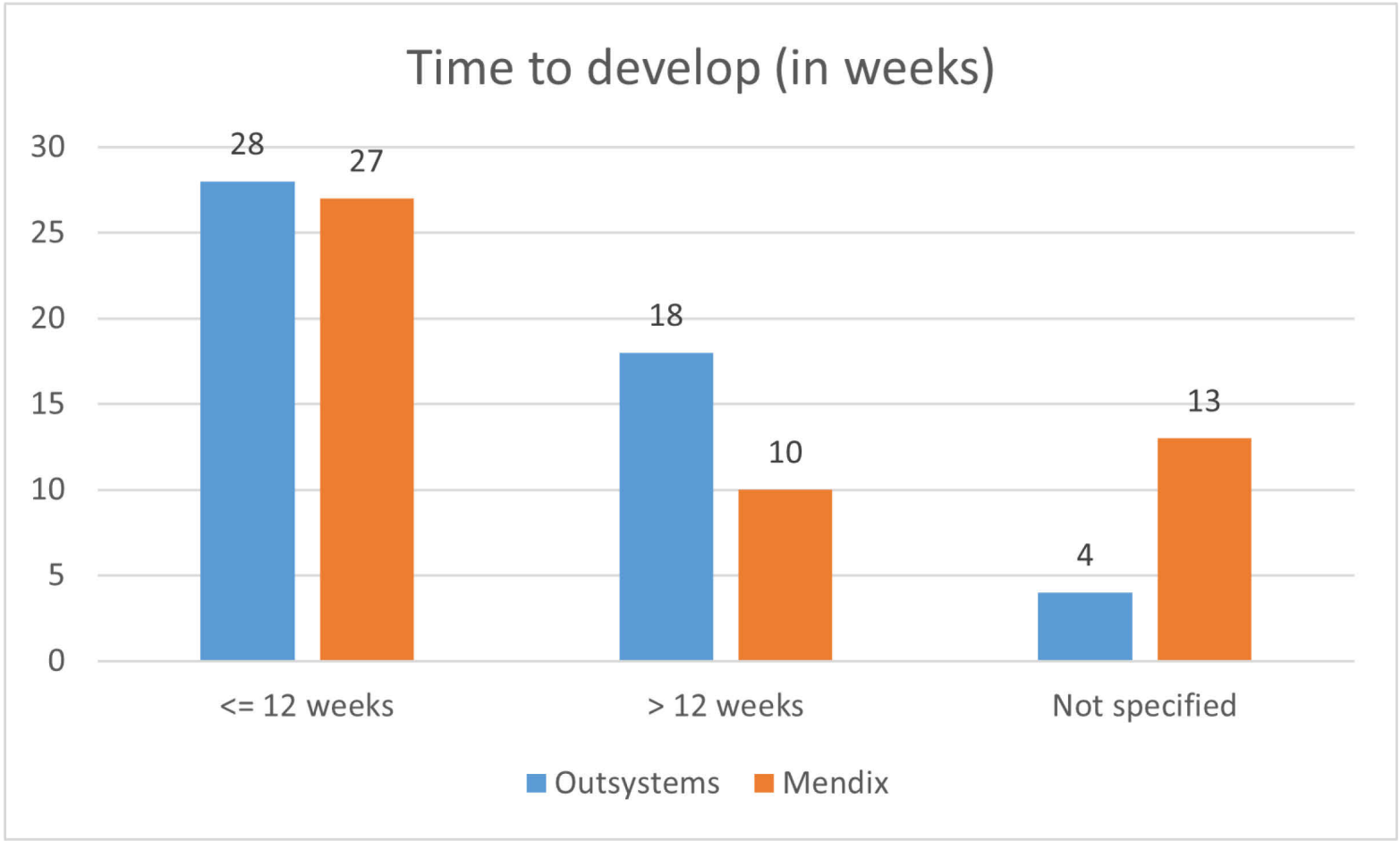
- Measured in weeks
- Threshold = 12 weeks
- Less than 12 weeks for 55/100
- More than 12 weeks for 28/100

Assumption:

- Low-Code enables faster application delivery (Forrester survey)
- Assumption confirmed!

Inference:

- Less development time → Experience-based test techniques
- More development time → Elaborate testing



Raquel Sanchis, ´Oscar Garc´ıa-Perales, Francisco Fraile, and Raul Poler. Lowcode as enabler of digital transformation in manufacturing industry. Applied Sciences, 10(1), 2020.

Scalability

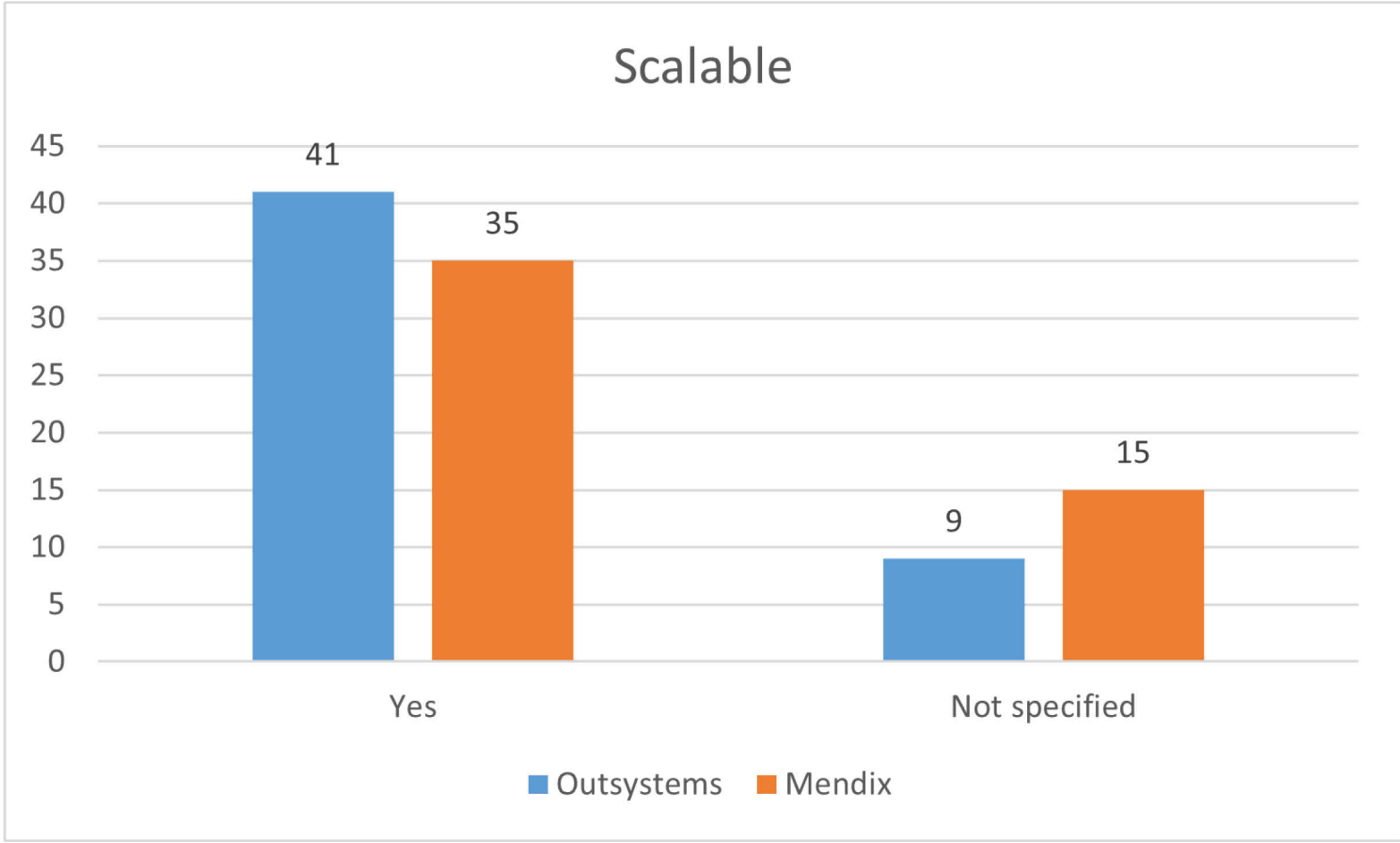
- Yes for 76/100
- Not specified for 24/100

Assumption:

- Low-code applications are developed mostly simple, non-scalable.
- Assumption proven wrong!

Inference:

Need of professional tester for performing non-functional tests.



Integration with External Systems

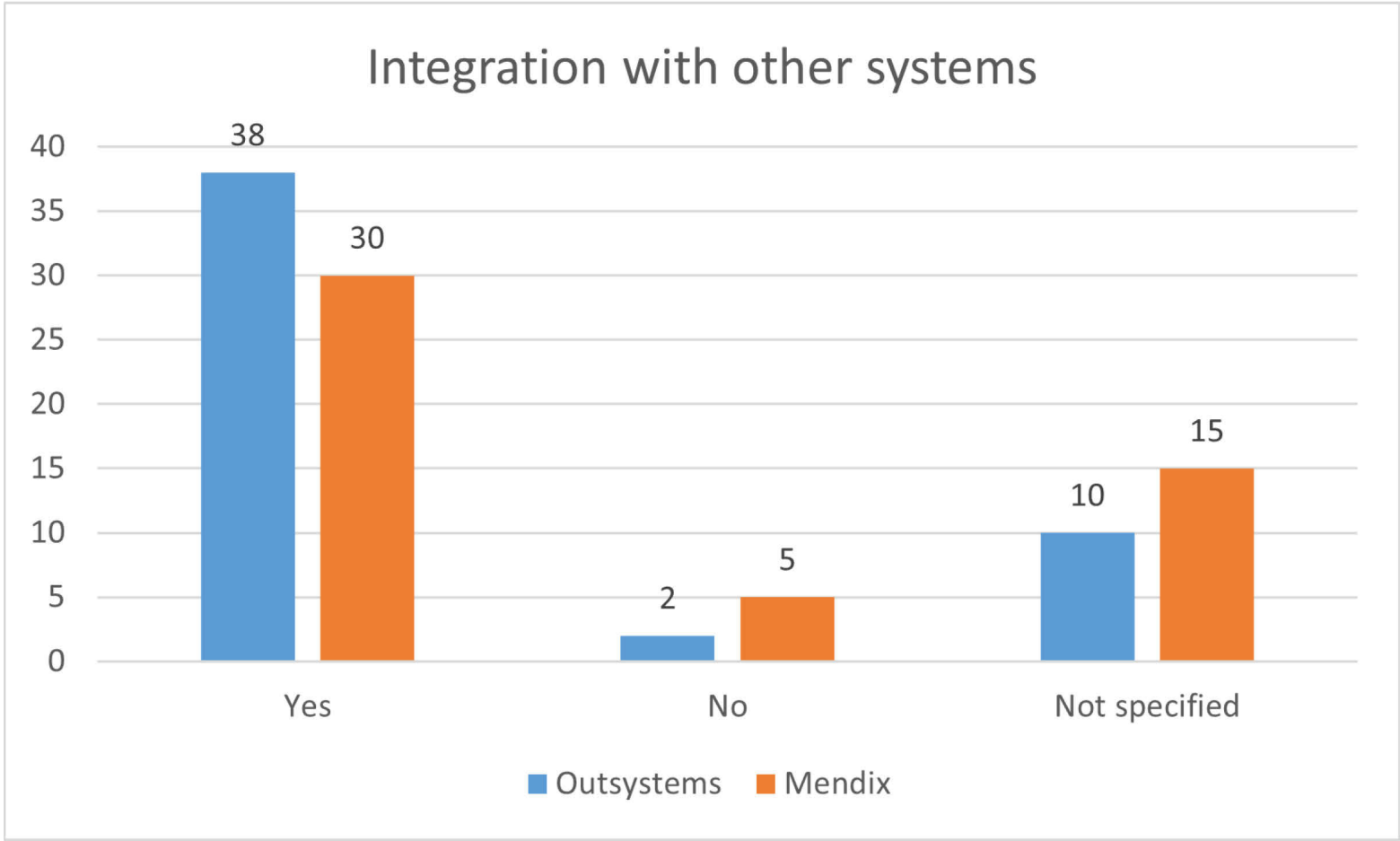
- Yes for 68/100
- No for 7/100

Assumption:

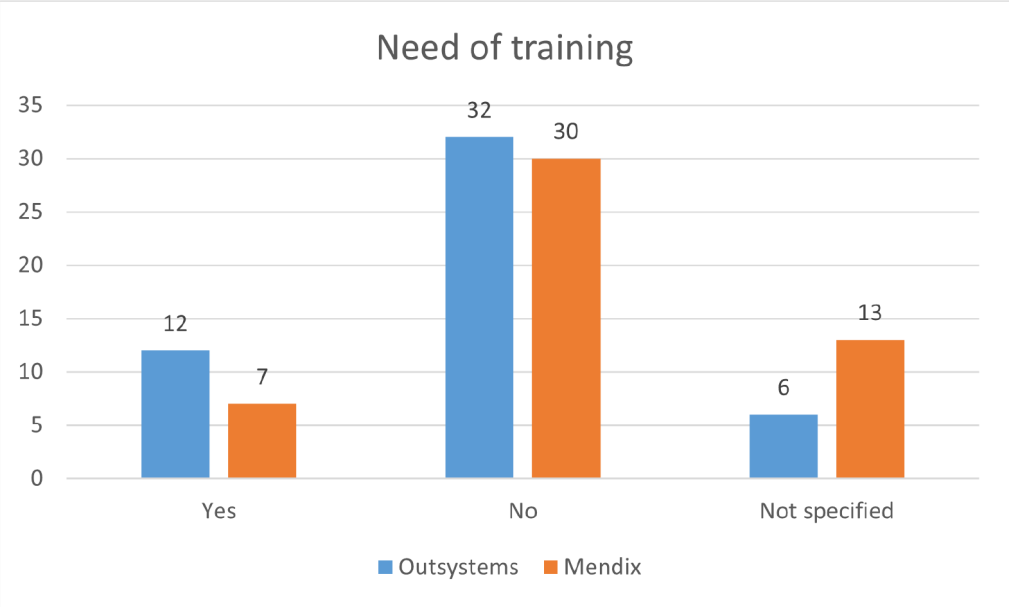
- Integrations with external systems may be difficult with Low-Code
- Assumption confirmed!

Inference:

There may be a need of all test levels – unit, integration, system, acceptance.



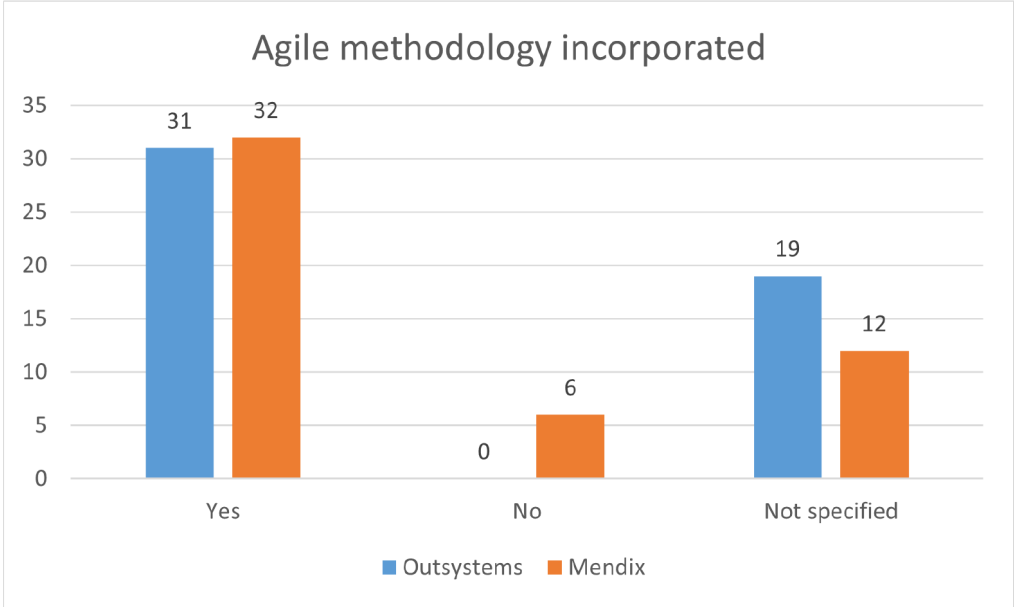
Need of Training & Agile Methodology



- Yes for 19/100, No for 62/100

Assumption for need of training:

- Low-Code platforms are designed specifically for 'citizen developer'
- Thus, most will need training.
- Assumption proven wrong!



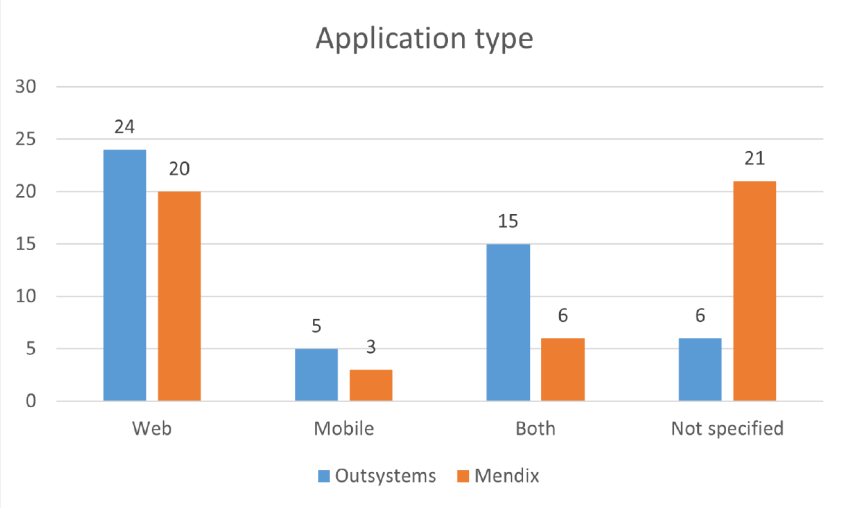
- Yes for 63/100, No for 6/100

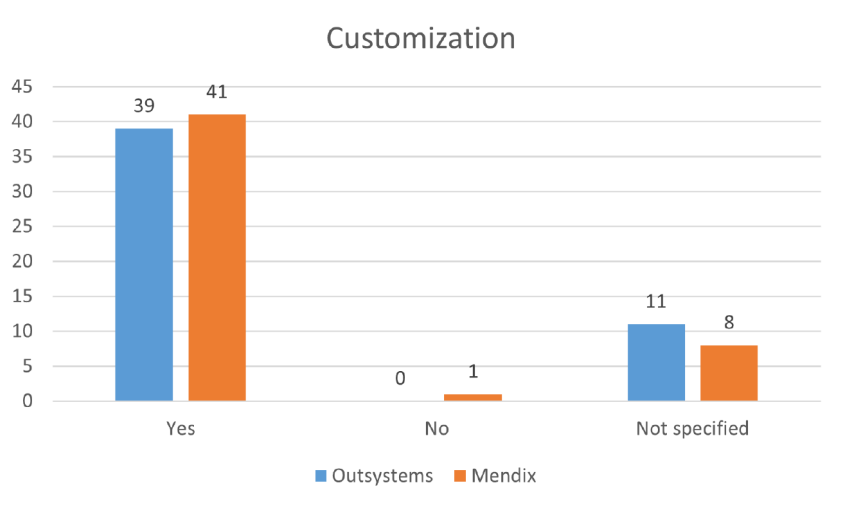
Assumption for agile methodology:

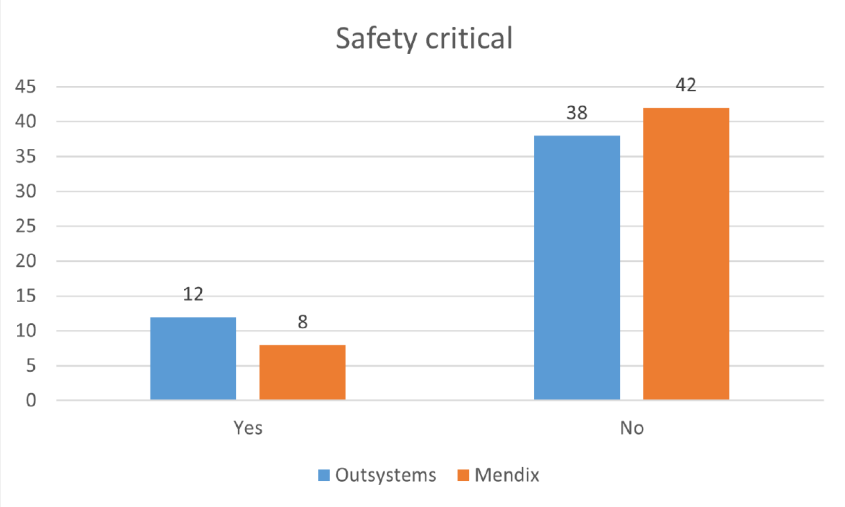
- Exploratory tests can be useful to test only changes in each sprint.
- Assumption incorrect !

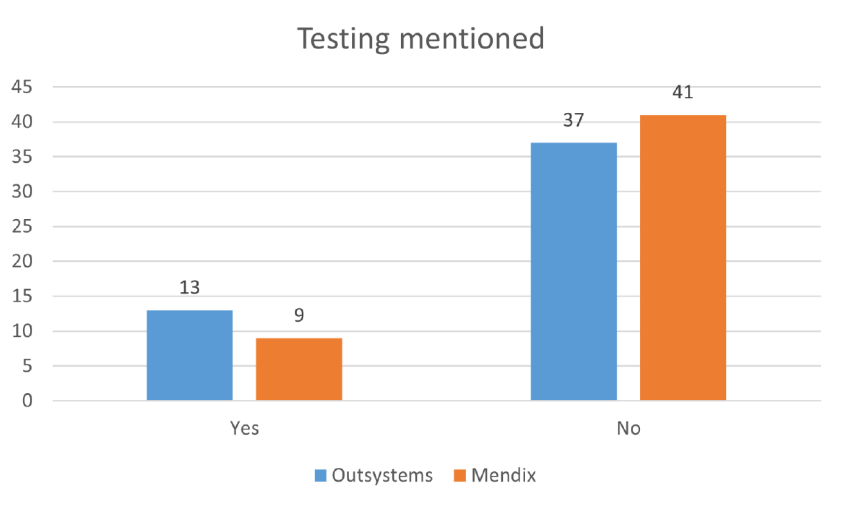
Information about test artifacts –
User stories for agile methodology

Other Complexity Characteristics









Rule Set for Test Methodology

Sr. No.	Questions	Test Aspects	Option 1	Option 2	Your Answer	Recommended Test Methodology
1	Do the requirements exist?	Test Techniques	Yes	No	Yes	Test Techniques : Black-box testing
2	Does the code exist?		Yes	No	No	Test Techniques : Experience-based testing
3	How much is the estimated development time?	Test Strategy	Less than / equal to 12 weeks	More than 12 weeks	1	Test Strategy : Analytical, Exploratory, Checklist-based on the requirements document
4	Is there integration with other systems?	Test Levels	Yes	No	No	Test Levels : Unit, System test
5	Is the application scalable?	Test Types	Yes	No	No	Test Type : Functional tests
6	Is a professional developer involved?	Test Roles	Yes	No	Yes	Developed by : Professional developer
7	Is professional testing necessary?		Yes	No	No	Tested by : Citizen tester
8	Is it an agile project?	Test Artifacts	Yes	No	No	Test Artifacts : Requirements, design document
9	Is test documentation necessary for any compliance/audit formalities?		Yes	No	No	Test cases : Test cases / test scripts need need not be documented

Evaluation and next steps

Evaluation by Testing of a Low-Code Application

- Digital notification board for technology center
- Test methodology determined & executed (see slide 11)
- Evaluation
  - Tasks/Responsibilities according to training: Development by professional developer (could be citizen developer), testing by – citizen tester (2-man principle)
  - Coverage of requirements: Out of 16, 4 were not implemented & 2 are not working
  - Defects found: 8 Defects, 2 Usability issues, 1 Security issue found
  - Testing aspects covered: test levels, test types, testing techniques, test strategy, test roles and test artifacts.
  - Time for testing (T_T) in proportion to time for development (T_D): $T_T = 10\% \, T_D$

Low-Code testing vs. Model-based testing

- Similar scenarios identified based on need for redundancy
- See master thesis for detailed scenario descriptions

Thank you
Any further questions?

Contact me:
Shreyasi.Warunkar@sn-cqm.de